Integration of Digital Video into Distributed Hypermedia Systems

Diplomarbeit in Telematik

Bernhard Marschall

März 1995

Begutachter: o.Univ.-Prof. Dr. phil. Dr. h.c. Hermann Maurer

Betreuer: Dipl. Ing. Dr. techn. Frank Kappe

Technische Universität Graz

Institut für Informationsverarbeitung und Computergestützte neue Medien (IICM)

Abstract

This thesis discusses digital video and its integration into hypermedia systems. It shows some algorithms and formats to compress and store digital video, focusing on the MPEG standard.

General topics on hypertext and hypermedia systems are discussed as well as their problems and drawbacks, and ways to overcome them. As a special example the architecture and features of Hyper-G and its Unix/X11 client Harmony are described.

Finally, the features and implementation of the Harmony Film Player are described. It fully integrates MPEG movies into the hyperlink structure of Hyper-G/Harmony by allowing links from and to movie documents. Any part, both temporal and spatial, can be used either as source or destination anchor of a hyperlink.

Contents

1. Introduction 6

2. Colour Spaces & Compression 9

2.1. Colour Spaces 9

2.1.1. The RGB Colour Space 9

2.1.2. The YUV (YC_bC_r) Colour Space 10

2.2. Compression 11

3. Digital Video 13

3.1. Motion-JPEG 13

3.2. MPEG 15

3.2.1. MPEG System 16

3.2.2. MPEG Video 16

3.2.3. MPEG Audio 20

3.2.4. Extensions of MPEG (MPEG-2, MPEG-3, MPEG-4) 20

3.3. H.261 21

3.4. Quicktime 22

3.5. Video for Windows (RIFF / AVI) 23

4. Hypertext and Hypermedia 25

4.1. What is Hypertext? 25

4.2. History & Applications of Hypertext/Hypermedia 26

4.3. Architecture of Hypertext/Hypermedia Systems 27

4.3.1. Structure of Hypertext Systems 27

4.3.2. Documents 27

4.3.3. Links 28

4.4. Navigation and Information Retrieval 28

4.5. Digital Video in Hypermedia Systems 29

5. Hyper-G 31

5.1. Navigation and Information Retrieval in Hyper-G 31

5.2. Architecture 32

5.3. Interoperability 34

6. Harmony 35

6.1. Architecture 35
6.2. A Tour through Harmony 35

7. The Harmony Film Player 45

7.1. Using the Film Player 45
7.1.1. The Control Elements 45
7.1.2. The Menu and Button Bars 47
7.1.3. The Progress Indicator 49
7.1.4. Dialogs 49
7.1.5. Selecting and Following Anchors 50
7.1.6. Defining and Deleting Anchors 51
7.1.7. The Film Player's X-Resources 54
7.2. Implementation Details 55
7.2.1. The MPEG Decoder 55
7.2.2. Encapsulating the MPEG decoder 58
7.2.3. The Harmony Communication Protocol 59
7.2.4. Implementing the Communication - The Dispatcher 62
7.2.5. The Frame List 63
7.2.6. Anchors 64
7.2.7. The User Interface - InterViews 69
7.2.8. Embedding the MPEG decoder into InterViews 70
7.2.9. The Viewer Class 71

8. Summary 75

Appendix A: The Harmony Colormap 77

References 79

1. Introduction

Information and its exchange has become an important part of our society. Communication networks arise everywhere around the world, most known and most widespread the Internet. The number of participants of the Internet has reached 20 - 30 million and keeps rising, and now covers private persons as well as universities and companies.

But with the enormous success and growth of the Internet, also its disadvantages have become evident: on the one hand, the individual is overwhelmed by the amount of data in the net, on the other hand, one can hardly keep track of the information provided; in short, it is nearly impossible to find exactly the relevant information.

To overcome these problems, distributed information systems have been developed, namely the ***Wide Area Information System (WAIS)*** [Stein91], which offers a set of search facilities over the WAIS servers and ***Gopher*** [Alberti92], which structures the information in a hierarchical way, very similar to the way files are structured in a file system of a modern operating system.

A third system has become very popular recently, the ***World Wide Web (WWW, W3)*** [Berners92], which was developed at CERN in Geneva, where it was used among physists to store and distribute their data. In the meantime, almost all institutions, research institutes, universities, as well as trading companies, restaurants, and others have their own WWW-servers to present their work or to offer their services. WWW is essentially a hypertext system, which means, text documents are connected by links (which are similar to cross-references in traditional books). External viewers are used to display other kind of documents, such as images, digital videos or audio.

Nevertheless, the existing systems still have their drawbacks: WAIS allows to search the database, but does not try to structure the data, Gopher on the other hand provides no real searching mechanism, and WWW suffers from the typical "lost-in-hyperspace" syndrome of hypertext: users do not know, how much information exists to a certain topic, how much they have already seen and where in information space they currently are.

So a very ambitious project was started at the Institute for Information Processing and Computer-supported New Media (IICM) of the Graz University of Technology (TUG): ***Hyper-G***, the first "second generation" hypermedia system [Andrews94b]. It takes the hypertext structure of WWW, but expands it in three essential points: first, it structures the data in a gopher-like way in so called collections, second, it provides a universal search mechanism on the whole database, and third, it is not restricted to texts, but fully supports other types of documents, such as images, films, audio, and others, as well as links between any type of documents.

Although Gopher and WWW clients can be used to retrieve data from a Hyper-G server, only native Hyper-G clients make use of the full functionality of the server. The most advanced Hyper-G client up to now is ***Harmony***, which runs on Unix machines under the X-Window

system. For each type of document (text, image, film, audio, postscript, ...) there is a separate viewer, which visualises these kind of documents.

One of these multimedia data types is digital video. Digital video becomes more and more important, as new ideas and concepts for television arise. These concepts are also strongly supported by the entertainment industry and aim towards a new, "individual" kind of television. Most known and discussed are:

- ***Video on Demand (VoD)***: a service provider sets up a video server; that is essentially a large database storing motion pictures in digital form. The television viewer can select any of these movies, which is then transmitted over some communication network and shown on the customer's television set.

- ***Interactive Television***: Interactive television goes one step further: the television viewer doesn't have to watch a movie passively, but can effect the flow of action. To implement interactive television, existing prototypes store various parts of a movie, which correspond to different flows of action. The viewer can choose among them by some interaction mechanism.

- ***Three-Dimensional Movies***: In a 3D-Movie the viewer can move through a scene, look into arbitrary directions, or zoom to certain points, thus pushing television towards virtual reality. Apple is going to release Quicktime VR (for Virtual Reality) this year, which will implement 3D-Movies as an extension to Quicktime (Apple's format for digital video).

When one wants to store or transmit digital video, ***compression*** becomes inevitable. A full colour picture requires 3 bytes per pixel; the PAL standard in Europe requires 25 frames per second, the NTSC standard in the USA requires 30 frames per second, yielding to a total data volume of 75-90 bytes per second and pixel. A television frame has about 352x288 pixels, which gives a total data rate of about 7.25 MByte per second. Different image and video compression algorithms and standards have been developed, among them MPEG (for Moving Pictures Expert Group), an international ISO-standard for compressing and storing digital video with associated audio. Although image (and more video) compression is of a high computational complexity, it becomes more and more used even with PC's, as hardware implementations of both the compression and the decompression algorithm become available.

Digital video is not only important as a replacement or further development for television, but also an interesting feature for new information systems. A known proverb says "A picture's worth more than thousand words". Of course, this holds even stronger for moving pictures. For example, short film clips can effectively be used to illustrate technical processes or present "virtual walks" through scenes. What makes them even more useful is their full integration into a hypermedia system's link structure, making parts of a film a source or destination of "cross-references".

This thesis discusses how digital video has been integrated into Hyper-G and its client Harmony: First, some general topics about compression and image and video compression in particular are discussed. Then the most common formats to store digital video, Motion JPEG, MPEG, H.261, Quicktime and AVI, are described, with the emphasis on MPEG.

After that hypertext and hypermedia, their history, areas of application as well as their problems are presented. Hyper-G as one existing hypermedia system, its architecture and features are described. Harmony, the Unix/X11 client for Hyper-G is discussed as well as the

main theme of this thesis: the Harmony Film Player, which integrates MPEG movies into the Hyper-G/Harmony environment.

When integrating a new type of document into a hypermedia system the most important question is how to integrate it into the hyperlink structure. In theory a link could be attached to any spatial and temporal part of the movie (called its source anchor). Real implementations have to be more restrictive: The Harmony Film Player allows users to define shapes (i.e. rectangles, circles, ...) in so called key frames. The shape in frames between two key frames is interpolated (either linear or as a quadratic B-spline). The first and the last key frame also define the temporal region of the anchor.

2. Colour Spaces & Compression

2.1. Colour Spaces

Colour is a subjective impression of individuals. A colour space is an abstract model, how colour can be "measured". Such an objective description of colour is the base to store and exchange colour data. There exist a lot of different colour spaces [Hill90, Color94]; each provides a somehow different view of colour. In the context of image compression and digital video, the most important colour spaces are the RGB- and YUV-colour spaces.

2.1.1. The RGB Colour Space

The RGB colour space is an additive colour space, i.e. each colour is described as a linear combination of the three primary colours red, green and blue (hence its name). Each colour is therefore given by a three-dimensional vector (R,G,B) (where R, G, B $\in$ [0;1]). The set of all describable colours (the colour space) forms the unit-cube; this RGB-cube is shown in figure 2.1.

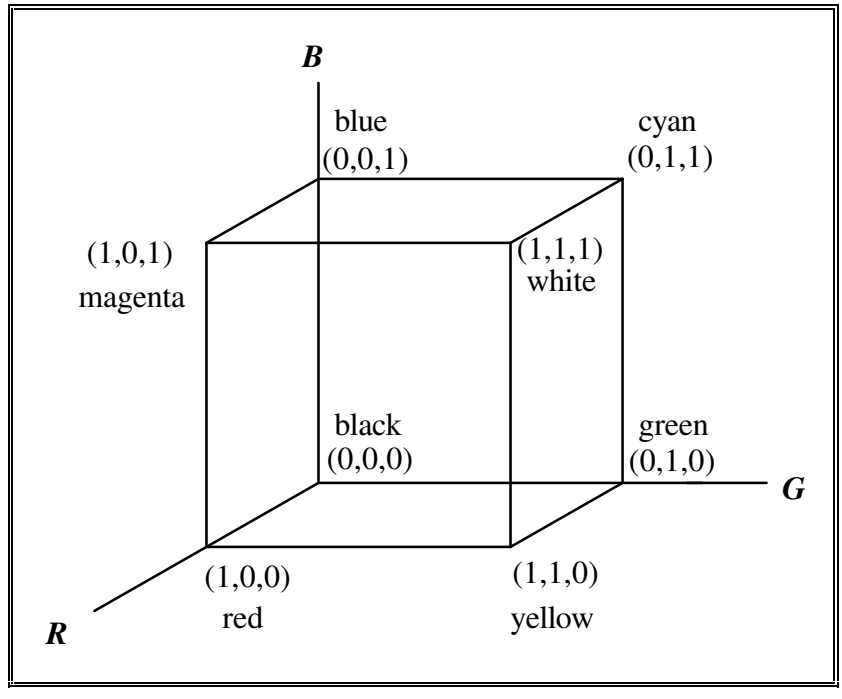

Figure 2.1: The RGB Colour Space

Almost all colour screens today are raster displays, which use a red, a green and a blue glowing phosphor, i.e. the RGB colour space, to display colours.

Normally, computers use one byte to store each colour component; the possible values for R, G, B are therefore not [0;1] but {0, 1, 2, ..., 255}. That gives a total number of 256^3 = 16.777.216 colours, which can be displayed on a colour screen.

If all these 16 million colours can be displayed on the screen simultaneously, the screen is called a ***True-Colour-Device***. True colour devices must store 3 bytes per pixel in their frame buffer.

Other screens cannot display all these 16 millions colours, but only a smaller number - for instance 256 - of them simultaneously. They are called ***Pseudo-Colour-Devices***. Pseudo colour devices use one or more colour maps, where the actual colour (the three bytes describing the red, green and blue component) are stored; the frame buffer contains only an index to this colour map.

2.1.2. The YUV (YC_bC_r) Colour Space

This colour space describes a colour by its luminance Y (the luminance signal) and two colour difference signals, called U and V (the chrominance signals). It is used by PAL, the standard for analogue transmission of colour TV in Western Europe.

The conversion from the RGB colour space to the YUV colour space is as follows:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$
$$U = B - Y$$
$$V = R - Y$$

Note that U and V can be negative. Therefore a variant of the YUV colour space, the ***YC_bC_r colour space***, is used in digital systems: C_b and C_r correspond to U and V respectively, but they are scaled and translated so that their values are in [0;1], as are the values of R, G, B and Y:

$$C_b = U/1.402 + 0.5$$
$$C_r = V/1.772 + 0.5$$

So we get the conversion from RGB to YC_bC_r colour space:

$$Y = 0.299 \cdot R + 0.587 \cdot R + 0.114 \cdot B$$
$$C_b = (B - Y) / 1.402 + 0.5$$
$$C_r = (R - Y) / 1.772 + 0.5$$

and back from YC_bC_r to RGB:

$$R = Y + 1.402 \cdot (C_r - 0.5)$$
$$G = Y - 0.34414 \cdot (C_b - 0.5) - 0.71414 \cdot (C_r - 0.5)$$
$$B = Y + 1.772 \cdot (C_b - 0.5)$$

The advantage of this colour space lies in the fact that the luminance and the chrominance signal, which are perceived differently by the human eye, can be treated differently by an encoding algorithm.

2.2. Compression

Compression algorithms in general, and image compression algorithms in particular, can be divided into lossless and lossy algorithms.

Lossless algorithms loose no information during the encoding process; that means, if an image is compressed and decompressed, the reconstructed image is exactly the same as the original image. Examples for lossless algorithms are huffman coding, runlength coding or the LZW-algorithm [Welch84]. These algorithms reach compression ratios of about 2:1 - in general their compression ratio depends on the input data.

Lossy algorithms, on the other hand, loose information during encoding; that means that the reconstructed image is not exactly the same as the original image. The art of lossy compression is to discard that information, which the human eye can not perceive. Lossy algorithms reach much higher compression ratios than lossless ones, up to 100:1. Examples of lossy compression algorithms are the Color Cell Compression (CCC) [Pins91], JPEG [ISO10918, Wallace91] or MPEG [ISO11172, LeGall91].

Compression for digital video can further be divided into intra- and interframe compression. In ***intraframe*** compression each picture is encoded independently of any other, just as it were a still image. ***Interframe*** compression uses the dependencies between pictures to gain higher compression ratios.

The aim of image compression, in general, is to reduce redundancy in the input data; images (as well as digital video) essentially contain three types of redundancy [Gonzales92]:

- ***Coding Redundancy*** is redundancy due to non-uniform probability distribution of the code symbols. It can be reduced by runlength or huffman coding.

- ***Interpixel Redundancy*** is due to dependencies, both spatial and temporal, of neighbouring pixels. It can be reduced using difference coding techniques or transformations to frequency domain (discrete Fourier transform or discrete cosine transform).

- ***Psychovisual Redundancy*** is due to information, which is not perceived by the human seeing. Algorithms that reduce psychovisual redundancy are always lossy, and try to discard non-significant information.

Fidelity Criteria

When using lossy compression algorithms, fidelity criteria get important. After all one doesn't want to discard too much or too important information. There are two kinds of fidelity criteria:

Objective fidelity criteria are the mean square error (MSE) and the signal-to-noise ratio (SNR). Both are measures for the difference of the original and the reconstructed image:

$$\mathrm{MSE} = \frac{1}{M \cdot N} \cdot \sum_{i=1}^{N} \sum_{j=1}^{M} \left(O_{i,j} - R_{i,j} \right)^2$$

$$\mathrm{SNR} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} O_{i,j}^{\ 2}}{\sum_{i=1}^{N} \sum_{j=1}^{M} \left(O_{i,j} - R_{i,j} \right)^2}$$

where $O_{i,j}$ are the brightness values of the original image and $R_{i,j}$ are the brightness values of the reconstructed image.

Often objective fidelity criteria are not satisfactory, because they just give the numerical difference of the two images, but not how strong this difference is perceived by a human spectator. This is were ***subjective fidelity criteria*** cut in: here the quality of the reconstructed image is judged by a group of humans.

3. Digital Video

3.1. Motion-JPEG

JPEG (for Joint Photographic Experts Group) is an ISO standard for compression and coding of still images [ISO10918, Wallace91]. Motion-JPEG, or short MJPEG (which must not be confused with MPEG), is just JPEG used for digital video: that means each frame is intracoded, independently of other frames, using JPEG. JPEG defines a number of compression methods; the most important of them (and what is normally meant when talking about JPEG) is the baseline sequential codec.

A single component (i.e. grayscale) image is encoded in three steps:

First the image is split into blocks of 8x8 pixels. The brightness values are shifted from range [0; 2^p-1] to [-2^{p-1}; 2^{p-1}-1]. Each block is then transformed using the forward discrete cosine transform (FDCT):

$$F(u,v) = \frac{1}{4} \cdot C(u) \cdot C(v) \cdot \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos\frac{2(x+1)\pi \cdot u}{16} \cos\frac{2(y+1)\pi \cdot v}{16}$$

$$where: C(u) = \begin{cases} \frac{1}{\sqrt{2}} & if\ u = 0 \\ 1 & if\ u \neq 0 \end{cases}$$

where f(x,y) is the brightness value of the pixel (x, y) of the block; the F(u,v)'s are called DCT-coefficients. For decoding the inverse DCT (IDCT) is used:

$$f(x,y) = \frac{1}{4}\left[\sum_{u=0}^{7} \sum_{v=0}^{7} C(u) \cdot C(v) \cdot F(u,v) \cdot \cos\frac{(2x+1)\pi \cdot u}{16} \cdot \cos\frac{(2y+1)\pi \cdot v}{16}\right]$$

The DCT is similar to the discrete fourier transform, which implies that the DCT-coefficients represent a two-dimensional frequency spectrum. Since F(0,0) is proportional to the average brightness of the block $\left(F(0,0) = \frac{1}{8} \cdot \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y)\right)$, it is (in analogy to signal theory) called *DC-coefficient*; the other 63 coefficients are called *AC-coefficients.*

In the second step the DCT-coefficients are quantized using a quantization matrix Q(u,v) with 8x8 elements. What you get are the quantized DCT-coefficients $F^Q(u,v)$:

$$F^Q(u,v) = \text{IntegerRound}\left(\frac{F(u,v)}{Q(u,v)}\right)$$

The dequantization, needed for decoding, is the inverse function:

$$F^*(u,v) = F^Q(u,v) \cdot Q(u,v)$$

Note that this is the only step in the whole process where information is lost, since $F^*(u,v) \neq F(u,v)$ due to the round function. The aim of quantization is to discard information, which is not visible to the human eye. (For instance the human eye is much more sensible to the average brightness of a block (i.e. the DC-coefficient) than to very high frequencies.)

In the third step these quantized DCT-coefficients are entropy coded: At first the coefficients are arranged in zig-zag order (see figure 3.1).

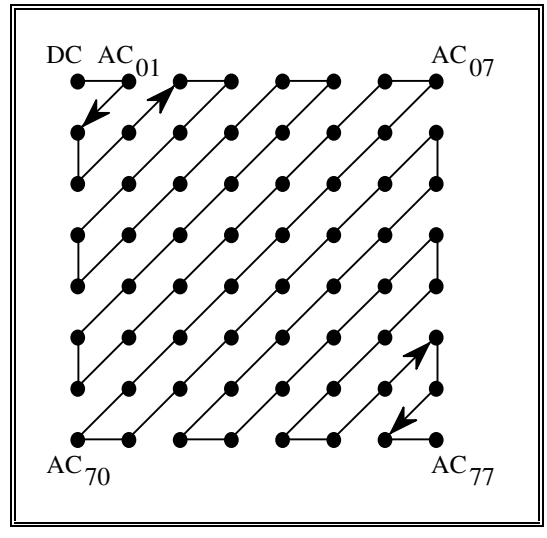

Figure 3.1: Zig-zag sequence

The DC-coefficients are differentially encoded; that means, not the current value of the DC-coefficient is stored but only the difference of the current DC-coefficient and the one of the previous block. Because normally the average brightness of consecutive blocks does not vary too much, the value of the difference is rather small and can well be compressed using huffman coding.

The sequence of AC-coefficients is runlength encoded as a pair of symbols s1 = (*runlength*, *size*) and s2 = (*amplitude*): *runlength* is the number of consecutive zero-valued AC-coefficients, and *size* is the number of bits used to encode *amplitude*, which is simply the value of a nonzero AC-coefficient. These symbols are finally huffman encoded using a table according to a probability distribution of the symbols.

JPEG handles colour images as so called multiple component images, where each primary of the colour space represents one component. The components need not have the same spatial resolution. Normally colour images are encoded using the YC_bC_r colour space,

were C_b and C_r are subsampled. Different quantization tables may be used for different components.

Figure 3.2 summarises the JPEG encoding process:

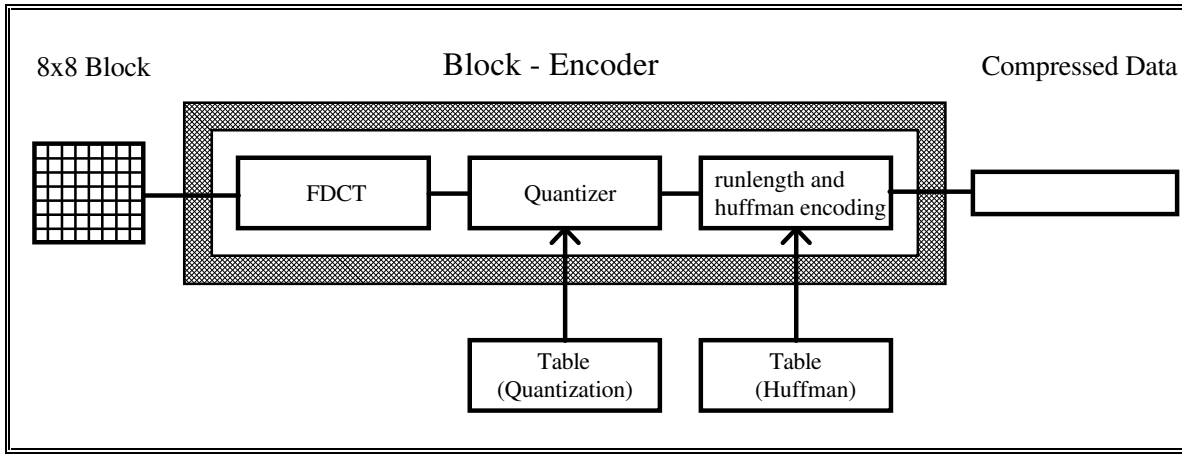

Figure 3.2: JPEG encoding process

MJPEG reaches compression ratios of 10:1 to 20:1 (i.e. 1-2 bits/pixel) without visible effects, 30:1 to 50:1 (i.e. 0.5-1 bits/pixel) with only moderate defects, and up to 100:1 (i.e. 0.25 bits/pixel) for low-quality purposes. This compression ratio is certainly not optimal, since the redundancy between two consecutive frames is not exploited (as is done in MPEG). Nevertheless MJPEG provides some advantages over MPEG:

- The JPEG committee started earlier than the MPEG committee, so the standardisation is further developed and hardware components supporting JPEG compression are well established.
- An MJPEG video can be more easily processed than a MPEG video since there are no interframe dependencies.

3.2. MPEG

MPEG (for Motion Pictures Expert Group) is also an international ISO-standard [ISO11172, LeGall91] which describes how to encode digital video with associated audio. The standard defines the syntax and semantics of the MPEG bitstream and hence the decoder. The encoder is not specified directly; so it is possible to adapt the encoder according to the needs of the application (real time compression vs. high compression ratio vs. high quality).

The bitstream is compressed to about 1.5 Mbit/s, which is also the bit rate of the (uncompressed) audio CD. The standard consists of 3 parts: part 1 describes the system coding layer, part 2 the encoding of digital video and part 3 the encoding of digital audio.

3.2.1. MPEG System

This first part of the standard describes, how video, audio, and private data streams are combined, synchronised and multiplexed.

An MPEG system data stream may contain up to 32 audio, 16 video, and 2 private data streams. An MPEG system stream is divided into packs. A ***pack*** may either contain system data (a system header) or one or more packets. Each ***packet*** contains an identifier, which gives its type and number (video 0-15, audio 0-31, or private), as well as a presentation time stamp, so that it can be displayed (or played) at the correct time.

3.2.2. MPEG Video

The encoding scheme for MPEG video is based on JPEG. The input data are the frames of the digital video in the YC_bC_r colour space. As shown in figure 3.3, the chrominance signals are subsampled: whereas the luminance value (Y) of every pixel is stored, the chrominance values (C_b, C_r) of 4 pixels are combined. This can be done without visible loss of quality because the human eye is much more sensible to luminance than to chrominance and it already halves the amount of data to encode.

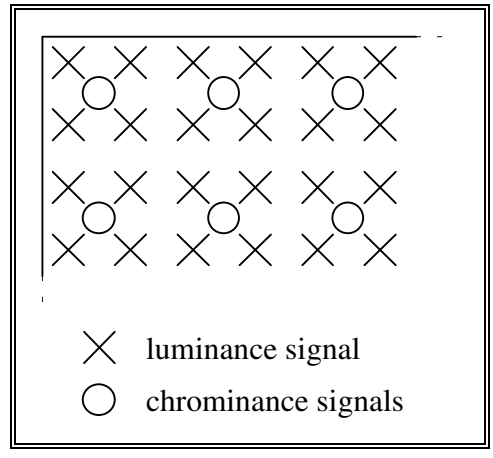

Figure 3.3: Subsampling of the chrominance signal

There are three types of frames: I-frames, P-frames, and B-frames. (There are also D-frames, which encode only the luminance signal in low quality, but they must not be mixed with other frame types. Video streams with D-frames are only used within system streams, and their only intention is to allow a fast forward search mode). Each frame is split into macroblocks of 16x16 pixel size; each macroblock contains 6 blocks of 8x8 brightness values (4 luminance blocks and 2 chrominance blocks).

I-Frames

I-frames (for ***intracoded frames***) are encoded just like still images using JPEG encoding, with some minor changes and simplifications:

The 6 blocks of each macroblock are DCT-transformed, the DCT-coefficients are quantized using a quantization table (in contrast to JPEG there is only one quantization table for both, the luminance and the chrominance signals) and finally the quantized DCT-coefficients are runlength and huffman encoded.

I-frames provide a starting point for decoding, since they do not depend on any other frame.

P-Frames

P-frames (for ***predictive coded frames***) depend on the previous I- or P-frame (called reference frame). Motion compensation is used to "predict" frames: for each macroblock, you search for the most similar region of 16x16 pixel in the reference frame (this so-called reference block need not be a macroblock of the reference frame, but can be any region of 16x16 pixels). The search can be done in one- or half-pixel resolution; the search algorithm itself is left to the encoder (the standard just suggests some reasonable algorithms).

What is really encoded is the prediction error: that is the difference between the current macroblock and the reference block. The same procedure as in I-frames is used to encode this error block: the prediction errors are DCT-transformed, quantized (with an other quantization matrix than the DCT coefficients of I-frames) and finally runlength and huffman encoded. Further you have to encode the motion vector (i.e. the number of pixels or half pixels in both directions by which the position of the current macroblock has to be translated to get the position of the reference block). These motion vectors are also differential encoded: you encode the difference between two consecutive motion vectors, not the motion vector itself. This is reasonable, because normally two consecutive blocks will be translated by approximately the same amount, so the differential motion vectors tend to be rather small and can be encoded efficiently using huffman coding.

If both the differential motion vector and the prediction errors are zero, the macroblock is encoded as ***skipped macroblock***. For skipped macroblocks no actual data are stored (only the macroblock header, identifying the macroblock as skipped macroblock), so they compress extremely well.

If the encoder cannot find a good reference block, the current macroblock is encoded as in I-frames. So P-frames can contain both I-blocks and P-blocks, whereas I-frames contain only I-blocks. It's entirely left to the encoder, which type of block to use.

B-Frames

B-frames (for ***bidirectional predictive coded frames***) are an extension of P-frames. Motion compensation is not only done for the previous (past) I- or P-frame but also for the following (future) one and the average of any past and any future block (it's up to the encoder which combinations it's taking into consideration).

So you have to encode a forward and a backward motion vector as well as the prediction errors. As in P-frames, if both motion vectors and the prediction errors are zero the block can be encoded as skipped macroblock.

Since there are now a lot more choices for the reference block, it is more likely that a really good one is found. Future prediction furthermore makes it possible that a good reference block can be found for previously hidden objects.

As in P-frames, if no good reference frame is found, the macroblock is intracoded.

Summary of Compressing Techniques used in MPEG

MPEG combines a number of compression techniques, which are here summarised:

- *Subsampling of the chrominance signal*: For 4 luminance values there is only one value of each chrominance signal; this can be done without visible effects, because the human eye is much more sensible to luminance than to chrominance.

- *Quantization*: A whole range of values is represented by a single value in that range. Since the range of possible values decreases, the number of bits needed to encode a value decreases. Quantization causes loss of information, which is called quantization noise.

- *Predictive Coding*: One tries to estimate the value of a pixel by values of previously encoded pixels; so only the difference between the estimation and the current value (the prediction error) has to be encoded. These prediction errors are normally small and can be encoded more efficiently. MPEG uses predictive coding to encode the DC-coefficients and the motion vectors.

- *Motion Compensation and Interframe Coding*: One tries to predict a block by a block of a known (reference-) frame. This technique relies on the fact, that within a short sequence frames don't change very much.

- *Frequency Transformation (DCT)*: Brightness values are transformed to frequency coefficients. In general the energy is concentrated in the lower frequencies, and most of the higher frequencies are zero.

- *Runlength coding*: If there is a series of equal values (i.e. zeros), one doesn't encode them all, but only their number and once the value.

- *Huffman Encoding (variable length coding)*: Highly probable values are encoded using only few bits, whereas improbable values get longer codes.

- *Picture Interpolation*: This is used in B-frames. One does motion compensation also with the average of a past and a future block, i.e. blocks that do not really exist in any frame of the sequence, but which are likely to be similar to blocks of the B-frame between.

Sequence of frames

The choice of the type of the frames used and their sequence, is entirely left to the encoder. An often used sequence is IBBPBBPBBPBB.... This sequence and the dependencies of the frames are shown in figure 3.4.

This sequence guarantees that there is an I-frame every 0.4 seconds (for the American TV standard NTSC with 30 frames per second) or every 0.48 seconds (for European TV standards PAL and SECAM with 25 frames per second) where decoding can start.

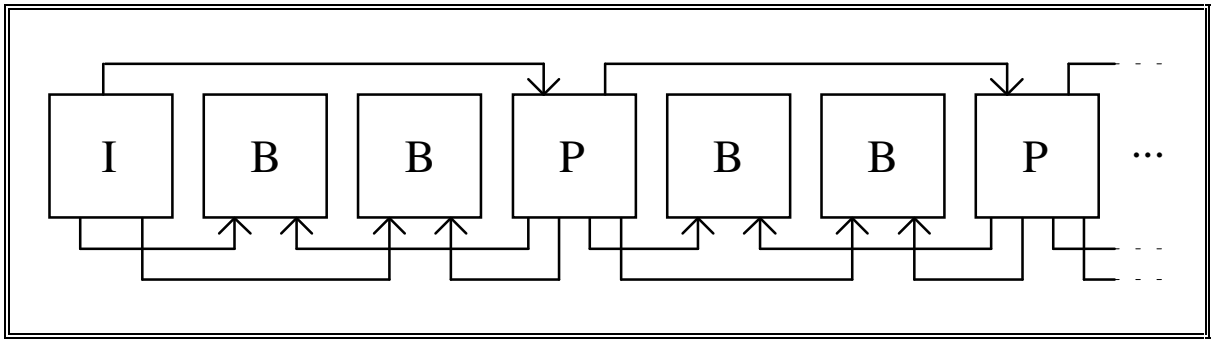

Figure 3.4: Dependencies of Frames

The motion prediction to future frames in B-frames rises one problem: to be able to decode a B-frame, you have to decode the future I- or P-frame in advance; therefore it has to be stored in front of the B-frame. So one has to distinguish between the display order and the file order of the frames.

The ***display order***

Offending Command = 11ont32
Error = nametype : undefined
this name is not defined in a dictionary
Stack =
707